소프트웨어의 확률적 신뢰도 성장법에 관한 연구

최규식[1], 김종기[2]
1)건양대학교 정보전자통신공학부, 2)건양대학교 첨단과학부

A Study on the Stochastic Reliability Growth of Software

Che Gyu Shik, Kim Jong Ki
Konyang University

Abstract - 소프트웨어 신뢰도에서 지금까지 여러 연구자들이 적용한 가정사항은 프로그램의 고장율이 잔여결함의 미지수에 대한 일정한 배수라고 한 것이다. 이는 모든 결함이 프로그램의 고장율에 동일한 양으로 기여한다는 것을 의미한다. 우리는 이 가정에 대해서 대안을 제시한다. 제안된 모델은 잔여결함을 중시하는 전의 것에 비하여 고장수정을 조기에 수행할 수 있게 함으로써 신뢰성 향상에 커다란 효과가 있다. 이 결함들이 전체적인 고장율에 가장 큰 공헌을 하기 때문에 그들 자신이 일찍이 나타나서 곧 수정될 수 있다는 장점이 있다. 모델은 취급하기가 쉬워서 다양한 신뢰도 척도를 계산할 수 있다. 목표 신뢰도를 얻기 위한 전체 수행시간과 목표 신뢰도를 얻기 위한 총 결함의 수를 예측할 수 있다.

1. 서론

소프트웨어 신뢰도에서 중요한 문제는 하드웨어에서 설계 오류를 줄이기 위한 여러 가지 조치들과 근본적으로 동일하다는 것을 알 수 있다. 이를 인식하여 두 가지 상황에서 유사한 수학적 모델을 사용하는 것을 연구하고자 한다.

만일 모든 결함이 제거된다면, 프로그램이 완벽하여 영원히 고장을 일으키지 않고 작동할 것이다. 이것을 고전적인 하드웨어 신뢰도 이론과 대비해보자. 하드웨어에서는 전체 시스템의 고장이 부품 고장에 의해서 발생한다. 이것을 볼 때 어느 시스템이 되었건 구성 부품 전체가 완벽할 수는 없기 때문에 시스템이 언제나 고장을 일으킨다는 것을 확신할 수 있다.

초기 연구에서는 동일크기의 일련의 단계에서 디버깅이 고장율을 향상을 이루게 하기 위해서 각각의 결함이 동일한 양으로 전체 고장율에 기여를 하는 것으로 가정했다. 프로그램이 N개의 결함으로 시작했다면, $(i-1)$번째 고장과 i번째 고장 사이의 수행시간 T_i는 지수적 분포를 한다.

이 상태에서 $(i-1)$개의 결함이 제거되었으므로,

$$pdf(t_i) = \lambda_i \exp(-\lambda_i t_i) \tag{1}$$

고장율은

$$\lambda_i = (N-i+1)\phi \tag{2}$$

라고 한다. 미지 파라미터 N과 φ는 최대 가능성(ML)으로 추정한다.

소프트웨어 신뢰도 모델링에 대한 초기의 연구에서는 초기의 포괄적인 하드웨어 이론으로부터 유도된 학습(lesson)을 강조하는 경향이 있었다. 현재 대부분의 연구자들은 하드웨어와 소프트웨어 사이에 부합되는 신뢰도 척도가 사용되어야 한다는 것에 동의하고 있으며, 그러므로, 복잡한 하드웨어/소프트웨어 시스템의 신뢰도 연구가 타당성이 있게 해야 진행되어야 한다는 것이다.

하드웨어 신뢰도 이론은 시스템 거동에서 부품고장의 원인에 초점을 맞추는 경향이 강했다.

소프트웨어 결함을 제거하는 모델들은 하드웨어 설계 오류상의 고갈을 적절히 나타낼 수도 있다. 제거할 수 없는 범주에 대해서는 고전적인 이론을 이용할 수 있으며, 결함을 제거하기 위해 여기에서 기술하는 것과 같은 새로운 신뢰도 성장모델을 이용할 수 있다.

2. 모델

시간-고장 분포에서 데이터는 고장 사이의 연속실행시간 t_1, t_2, t_3, …, t_i, …의 시퀀스가 된다. 단순하게 하기 위해서 결함제거가 순간적이고, 결함이 발견될 때마다 확실하게 제거되는 것으로 가정한다.

확률변수 T_1, T_2, …가 s-독립이라고 가정하자. 어떠한 시간에 고장율이 알려져 있다고 하면, 그 고장율에 따라서 고장이 무작위적으로 발생한다고 하는 것이 합리적일 수도 있다.

총 경과시간이 τ이고 i개의 결함을 수정한 경우를 생각해보자.(그림 2) 잔여결함수는 $N-i$ 개다. 결정적인 점은 이 $N-i$ 개의 결함들이 각각 다른 발생율 φ_1, φ_2, …, φ_{N-i}를 가질 것이란 사실이다. 프로그램의 고장율은

$$\lambda = \phi_1 + \phi_2 + \cdots + \phi_{N-i} \tag{3}$$

이다.

각 결함에 대한 고장의 시간분포가 지수분포라 가정하면, φ가 기지수일 때 프로그램의 현재 신뢰도를 완벽하게 기술할 수 있다. 그러나, φ는 미지수이다. 프로그램의 고장율은

$$pdf(t \mid \Lambda = \lambda) = \lambda \exp(-\lambda t) \tag{4}$$

로 한다.

(3)으로부터 프로그램의 고장율 Λ는 $(N-i)$개의 감마 확률변수의 합이므로, 아래와 같은 pdf를 가진다.

$$(\beta+\tau)\cdot gamd([\beta+\tau]\lambda;[N-i]\alpha) \tag{5}$$

3. 신뢰도 예측

본 장에서는 모델 파라미터 N, α, β가 기지수인 것으로 가정한다. 물론, 디버깅 과정동안 획득한 데이터로부터 이러한 파라미터를 추정하는 것이 중요한 문제이다. 이러한 형태의 모델들은 두가지 목적으로 쓰인다.

미래의 신뢰도 예측은 두 가지 방법으로 접근 가능하다. 한 쪽은 어떤 규정된 실행시간이 경과된 후에 프로그램이 얼마나 신뢰성 있게 작동하는가 하는 것을 알기 원할 때가 있다. 또 다른 한편으로 얼마만큼의 규정된 결함수를 수정한 후에 그 신뢰도가 얼마나 될까 하는 것에 좀더 관심을 가질 수 있다. 실행시간과 역일 시간이 거의 동일하다면 첫 번째 접근방법이 좀더 유용할 수 있다. 두 번째 접근법은 실행시간이 "수리"시간에 비하여 작을 때에 더 유용하다. 이 문제는 단지 이 모델이 "수리시간"을 무시하기 때문에 발생된다. 불행히도, 전체 역일 시간모델에 합쳐질 수밖에 없는 수리시간분포에 대해서는 합의된 바가 없다고 볼 수 있을 정도이다. 이 분

야는 앞으로 많은 연구를 필요로 한다.

3.1 신뢰도 k-고장

전체 경과시간이 τ이고 이 기간동안 i개의 결함을 발견하여 수정하였다. 우리는 지금 k개의 고장이 더 발생한 후 프로그램의 신뢰도가 어떻게 될 것인가에 관심이 있다. 우리는 $T(\equiv T_{i-k-1})$의 무조건적인 분포를 요한다.

$$pdf(t)=\int\cdots\int pdf(t|T_{i+1}=t_{i+1},\cdots t_{i+k}=t_{i+k}) \times pdf(t_{i+1},\cdots,t_{i+k})dt_{i+1},\cdots,dt_{i+k} \quad (6)$$

(6)은 일반적으로 봐서 해석적인 방법으로는 구할 수 없다. 고장율을 Λ라 하면.

$$\Lambda=[(N-i-k)\alpha]/[\beta+\tau+\sum_{m=i+1}^{i+k}T_m] \quad (7)$$

이러한 결론은 목표 신뢰도를 성취하는데 필요한 디버깅 노력을 평가하는데에 중요한 가치가 있는 것이다. 이러한 목표 신뢰도를 얻기 위한 여러 가지 방법이 있다.

3.2 미래의 실행시간 τ'가 경과한 후의 신뢰도

이제 그림1의 B 시점에 있는 프로그램의 신뢰도를 고려해보자. 전과 같이 전체 실행시간 τ동안 i개의 고장(그리고 따라서 i개의 결함을 수정)을 관찰하였다. 미래의 디버깅 시간(실행시간) τ'가 경과한 후 프로그램이 얼마나 신뢰성 있게 작동하냐 하는 것에 관심이 있다.

확률변수 T가 다음 고장까지의 시간을 나타낸다고 가정하자. (τ, $\tau+\tau'$) 기간동안 (N-i)개의 결함을 모두 제거하는 영이 아닌 확률이 있으므로, T가 무한인 영이 아닌 확률이 뒤따른다.

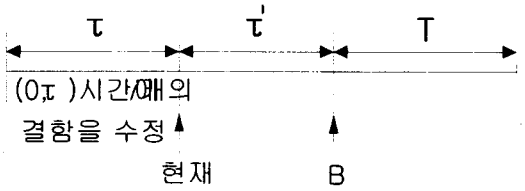

그림 1. 고장과 고장시간의 관계

$$F(t)=\sum_{k=0}^{N-i}\Pr\{T\le t|K=k\}\cdot\Pr\{K=k\} \quad (8)$$

여기서, 확률변수 K는 (τ, $\tau+\tau'$)에서 발생하는 고장의 수를 나타낸다. 이제 B점에서 ($i+k$) 결함이 수정되고 전체 실행시간이 ($\tau+\tau'$) 만큼 경과되었으므로,

$$\Pr\{T\le t|K=k\}=1-\left[\frac{\beta+\tau+\tau'}{\beta+\tau+\tau'+t}\right]^{(N-i-k)\alpha}$$
$$=parf(t/[\beta+\tau+\tau'];[N-i-k]\alpha) \quad (9)$$

이다.

비록 T가 어떠한 모멘트도 가지고 있지 않지만(특히 MTTF가 존재하지 않음) (34)로부터 B점의 고장율을 발견할 수 있다.

$$\lambda=R'(0)/R(0)=\frac{(N-i)\alpha(\beta+\tau)^{\alpha}}{(\beta+\tau+\tau')^{\alpha+1}} \quad (10)$$

이는 T의 무조건적인 분포로부터 구해지므로 확률변수가 아니다. 이는 τ'의 함수로 취급할 때 모델의 위험도이다.

3.3 마지막 결함을 제거할 때까지의 시간

한 가지 경우로서 마지막 결함을 제거할 때까지 즉, "완벽한" 프로그램을 얻을 때까지 필요한 디버깅시간(실행시간)의 분포를 아는 것이 큰 관심사일 수가 있다. 이 확률변수를 T^*라고 하자.

i개의 고장이 (0, τ)기간 동안 이미 제거되었으므로, 프로그램에는 (N-i)개만 남아 있다. 이러한 잔여 결함 중에서 시간 X_j가 지난 후에 j번째 결함이 고장을 일으켰고 또 그 결함이 제거되었다고 하자. 그러면

$$T^*=\max_{j=1,\cdots,(N-i)}\{X_j\} \quad (11)$$

그러나, X_j는 모든 j에 대해서 Pareto분포를 가진다. 이는 j 번째 결함의 고장율 Φ_j가 감마분포를 가지며,

$$pdf(x_j)=\int\phi_j e^{-\phi_j x_j}pdf(\phi_j)d\phi_j$$

이기 때문이다.

그러므로, T^*에 대한 cdf는

$$\Pr\{X_1\le t,\cdots,X_{N-i}\le t\}=[\Pr\{X_j\le t\}]^{N-i}$$
$$=\left[1-(\frac{\beta+\tau}{\beta+\tau+t})^{\alpha}\right]^{N-i} \quad (12)$$

(12)는 어떤 주어진 t 시간에 모든 결함을 제거하는 확률을 직접 계산하는데 쓸 수 있다.

한편으로, 주어진 확률 예를 들면 0.9를 가지고 모든 결함을 제거하기 위해 필요한 실행시간을 직접적으로 계산하는데 쓸 수도 있다.

3.4 주어진 기간 동안의 고장

고장에 의해서 정지되는 정지시간에 관한 것과 결합하여 관심사가 되는 기간 예를 들면 수행시간 또는 전 수명사이클에 걸친 시스템의 이용율에 관한 정보를 제공할 수 있다.

3.5 k개의 고장에 대한 실행시간

k개의 고장이 발견될 때까지 지금부터 전체 시간의 cdf를 고찰해보기로 한다. 이는

$$\Pr\{T_{i+1}+T_{i+2}+\cdots+T_{i+k}<t\}$$
$$=\Pr\left\{Z>\left[\frac{\beta+\tau}{\beta+\tau+t}\right]^{\alpha}\right\} \quad (13)$$

이고, 여기서, $Z^d\ B(N-i-k+1,k)$이다.

4. 모델파라미터 추정

일반적인 데이터 집합은 고장 사이의 연속된 실행시간 열 t_1, t_2, …, t_n이다. 추정하는데에 3개의 파라미터 N, α, β가 있으며, 이는 J&M모델에서 두 개의 파라미터 N, φ인 것과 비교가 된다.

$$pdf(t_m|t_1,\ t_2,\ \cdots,\ t_{m-1})$$
$$=pdf\left\{t_m|\text{전체 실행시간 }\sum_1^{m-1}t_i\text{에서 }(m-1)\text{개의고장}\right\}$$
$$=(N-m+1)\alpha(\beta+\sum_1^{m-1}t_j)^{(N-m+1)\alpha}/(\beta+\sum_1^{m-1}t_j+t_m)^{(N-m+1)\alpha+1} \quad (14)$$

이므로, MLE는

$$L(N,\ \alpha,\beta)=\prod_{m=1}^{n}pdf(t_m|t_1,\ t_2,\ \cdots,\ t_{m-1})$$

$$= \prod_{m=1}^{n} \frac{(N-m+1)\alpha(\beta+\sum_{1}^{m-1} t_j)^{(N-m+1)\alpha}}{(\beta+\sum_{1}^{m-1} t_j + t_m)^{(N-m+1)\alpha+1}} \qquad (15)$$

MLE 방정식은 N, α, β에 대해서 해석적으로 구할 수가 없다. 따라서, 수치해석적인 방법으로 구할 수밖에 없다.

5. 결론

여기에 제시된 모델은 이미 존재하고 있던 기존의 결함계수 모델을 비판하는 데에서 시작되었다. 그 의도는 기존모델의 심각한 반대 현상으로 나타나는 것을 극복하기 위한 것이다. 즉, 프로그램내의 모든 결함에 대해서 프로그램의 고장율 입장에서 동일한 심각성을 가진 것으로 가정한 것을 말한다. 사실상 결함의 심각성에 있어서 여러 다양성이 있으며, 이것이 결함제거로부터 연유되는 신뢰도 성장모델에 반영되어야만 하는 것이다. 여기에서 제시한 모델과 같은 모델과 기존의 결함 계수 모델이 프로그램(또는 하드웨어 설계)의 고장거동을 취급할 뿐이다. 어떤 고장들은 다른 것(발생율을 차별화하는 문제와 확연히 구분)보다 훨씬 더 심각한 결과를 가져오기도 한다. 그래서 우리는 이러한 비용(또는 결과) 공정을 모델링해야 했다. 이러한 관찰은 동일한 힘으로 고전적인 연구에 적용한다. 그러나, 설계오류를 연구할 때는 특별한 힘을 가지고 있다. 치명적인 설계의 고장영향으로부터 우리를 보호하는 능력은 극히 제한되어 있다. 이는 기존의 신뢰도 연구와 대비가 되며, 특별히 치명적인 부품중에 여유다중을 가지는 것이 가능한 경우에서 대비된다. 우리는 소프트웨어 및 설계 신뢰도 분야에서 유사한 기법을 요한다. 일부 공정을 여기에서 진행하였다.

본 연구는 한국과학재단 목적기초연구(2000-2-30300-001-2) 지원으로 수행되었음

[참 고 문 헌]

[1] Z. Jelinski, P. B. Moranda, "Software reliability research", in Statistical Computer Performance Evaluation, Academic Press, pp465-484, 1972

[2] M. Shooman, "probabilistic models for software reliability and prediction", in Statistical Computer Performance Evaluation, Academic Press, pp485-502, 1972

[3] B. Littlewood, "stochastic reliability-growth : a model for fault-removal in computer programs and hardware design", IEEE trasns. on reliability, pp313-320, 1981